# Parallel Computer Architecture

**Hemangee K. Kapoor**
**Department of Computer Science and Engineering**
**Indian Institute of Technology Guwahati**

**Week - 02**
**Lecture - 07**

Lec 7: A shift from sequential to parallel

Hello everyone. We are doing module 1 introduction to parallel architectures. This is lecture number 7 and in this lecture we are going to see a shift from sequential to parallel. Up to previous lecture we have seen performance, speed up, execution time, how to report the results. We have looked at Amdahl's law, Gustafson's law and so on. The conclusion was that we need parallel architectures and today we will again reinforce this understanding and look at how we move from sequential to parallel. Alright.

So we are building parallel systems but why? Because the single processor performance can no longer be sustained. Earlier it was defined with the transistor density that is the processor performance improved with the transistors with the number of instructions you could execute, the development in the instruction set architecture, the processor speed and then instruction level parallelism. All these factors improve the single processor performance. However, we saw that the frequency could not be increased beyond a point because of power constraints, because of heat issues and so on.

So the chips became too hot to be usable. The limits of heat dissipation had reached and we could not increase the clock frequency and therefore to increase the number of or rather to increase the usage of these transistor density we thought of adding parallelism and going multi-core. Okay. And we have seen some examples in the previous lectures and this slide also lists some newer processors which are multi-core from 6 cores to 70-80 core systems exist and GPUs definitely have several more processing elements. So what are the consequences of this multi-core revolution on computing? I would say the easy times have gone. Why? Because in the single core system we had better performing applications just because the processor was good.

I had a good ISA, a good frequency, nice ILP given to me by the computer architecture and the same program ran very fast. But now this easy way is not possible. Just updating myself or just updating my processor to a newer generation is not going to automatically improve the performance for us. And parallel computing techniques need to be exploited with every newer generation because we have every new generation is coming with more cores. Thus the parallel computing is going mainstream. Alright.

So why do we need to write parallel programs? Alright. I have a multi-core system but cannot I use it the same way I used to use a single core system. Alright. So programs that were written for a conventional single core systems cannot exploit the multiple cores because the program was compiled, keeping in mind that you have a single core. It has a specific ISA, instructions are scheduled in a particular manner and hence it only sees a single core. No matter your program is running on a multi-core system. Okay. So that program on a single core cannot simply exploit the multiple cores. We could do one thing that run multiple programs on these different cores.

That is multiple instances of the same program is one way out but does this help us? Of course I can run different programs on the multiple cores but if I want to improve the performance of a particular application I need to exploit the multiple cores for this particular application. Okay. For example you are writing a gaming application. Would you want to have multiple instances of the game run on the same processor or you want the gaming software to perform better? Definitely you want your game to run faster rather than run multiple games on different cores of the same system. Okay. So how do we do this? Rewrite single programs to go parallel so that we can exploit the multiple cores. That is we need to rewrite the sequential program into a parallel one.

Can this be automated? Yes, it can be to a certain extent but it is not always possible. There has been limited success in auto translation of serial code to parallel code because it is not very straightforward to identify complex constructs in a serial program because all our serial programs are becoming more complex and it is difficult to identify the constructs which can be chopped into small pieces to be run in parallel. Okay. So what do we do? Step back and devise newer algorithms for a parallel system. We have seen this example in the vector architecture, but a quick recap to understand the concept. If you want to sum the elements in an array in a sequential manner what will you do? Okay. So this is my array.

May have thousand, hundred, how many ever elements to be added. So I will just take the first element and add this element to the second one. So first do this pair of addition, then this result gets added to the second one sorry to the third element and then we take the fourth element and then add the fourth element to the sum of the previous. Getting it? So this is how I would do this on a sequential or a serial manner adding two elements at a time and accumulating the sum in the previous one. So this is the same sum getting accumulated.

So we are going to accumulate the answer and at the end of all this journey here is where you will get the final answer. Okay. So the final sum will come here. So you can see there is a cascade of all these operations sequentially done one after the other to compute the sum in series. The same thing if you want to do in parallel what will you do? I will chop the data into small pieces and then give that work to different multi-cores right ? every core will do the addition and every core definitely has to do a sequential addition similar to the previous one. So it does this and this way. Okay so this is sum S1 and similarly here you will get through a tree of additions you will get an S2, S3 and S4. Okay.

Eventually our job will be to add these. Okay. So to find some of the elements in parallel, I will chop the work across multiple cores. In this example I have taken more smaller pieces, every piece having only two elements at a time so only two elements, two elements here, two elements here. So the blue range of elements gives me this yellow range of elements which is the sum of pairs of the numbers. Then we add the two yellow ones to get the pink one and then we add the pink ones to get the green one and then the blue one which is the last one all right. So with this tree structure we are able to generate the final answer.

So what lesson do we learn from this example, that is the sum of elements of an array. So it is tough for a translator to discover the procedure. Okay. Certain softwares are able to identify the common serial constructs and parallelize them. However with increasing complex serial applications and serial programs it is difficult to auto translate or auto divide these programs. So therefore we cannot continue to write serial programs, we must write parallel programs to exploit the multiple cores. So how do we write parallel programs? This was an example but is there a proper method to do this? So to write a parallel program you have to divide the task and dividing the task comes under two different types, one is task parallelism and one is data parallelism.

So you have to divide the tasks across the processors or you have to divide the data across processors. okay. So let us do this with an example, okay. So the example is about checking answer books after an examination. So we have Professor P and the professor has got four TAs A, B, C and D okay. So there are five resource people to do the job and what is the task to be done? We have hundred answer books to check and each having five questions, okay.

So we have how many resources? We have five resources and what is the task? We have to check questions, five questions and there are how many answer books? Hundred. okay sorry hundred answer books. So this questions I can say I have question 1, question 2, 3, 4 and 5. I have five questions to check. So we have stack of answer books kept here to be checked. So several answer books are there and each answer book has got five questions to check.

So how do you divide the task? One option is task parallel. So what is task parallel? You have to define what a task means in your current problem statement. So it is up to us how we define it because at times there could be multiple options available. In this case I will say that every question is a task for me, okay. So task is question wise.

So each question is one task. So how many tasks do we have in this application? We have five tasks in this application and I can give one task to every resource. So with this definition what will you do? You will give question 1 to resource 1, question 2 to resource person 2 and so on okay. So what does this mean? When resource person 1 gets question 1, this person is going to check question 1 of all the answer books starting from 1 to 100. So they will check book 1, question 1, book 2, question 1 and so on.

So resource person 1 is going to check question 1 of all the answer books, resource person 2 will do question 2 of all the answer books and so on. So this is how we can exploit task level parallelism. Now I think you must have already concluded what would be data parallelism in this example, okay. So you can pause the video, think for yourself if you have not decided the method. Alright. So what is remaining? What is the other way of thinking? We have these answer books, right?

So for data parallelism I can use the answer books. We have 100 answer books and if I treat the answer books as data, okay. So if I treat this as data, I have to divide the data across five resource persons and how are we going to do that? I will say my data is the answer book and this answer book has to be distributed across the resource persons. So R1 will get 20 books and so on, resource person 5 will get the remaining or last 20 books. So every resource person gets 20 books to check and when they get the 20 books they do the complete question set.

So question 1 to question 5, all questions are checked by the same resource person on this set of data. So we have divided the data parallely across the resources and every resource is doing the complete program execution that is checking question 1, 2, 3, 4, 5 on his or her set of data. So that is data parallelism, okay. So task parallelism give one question each to a resource person and data parallelism give 20 answer books to every resource person to check, okay. So that is how one can think of moving from a serial to a parallel and dividing the tasks.

So in theory can we evaluate how fast we will go? So on the left hand side I have given an example if you want to sum n numbers okay. So summing n numbers, so to add n numbers on a sequential machine you will need , $O(n)$ complexity, you want to add pairs at a time where only one number can be handled in one time unit, okay. So $O(n)$ is the complexity of adding n numbers. You want to sort n numbers, yes you are going to take order $O(nlog\, n)$ time on a sequential machine. Now the same task if I want to do in a parallel machine, if you see here the previous slide had an example. We drew a tree and the tree height was log n so that would decide the time complexity.

So time to sum n numbers was log n in a parallel system where n is the number of processes available sorry n is the number of numbers you want to add, okay. Then time to sort n numbers, well if you think I want to just divide this by n it is not actually log n but the best we could get is log n. So that is in theory but in practice we are not going to get such a good speed up. Why? Because in practice there are many constraints we need to follow, because first thing is we need to define how the data is distributed across the processes, right? We are going to divide the workload and so the data will be given to different cores.

How is the data going to be named in this? Then how are the processors going to communicate with each other, coordinate and synchronize with each other? So these are the problems we need to handle. Then selecting the processing node size, how big a processing node I should select? should it be a complete personal computer or can it be a small ALU or a small streaming multiprocessor like in a GPU. So what is the size of my processing element? and how many such processing nodes I need to have in my system? So theory, I can say that I can theoretically get this much speed up but in practice all these questions need to be answered. So if we look at the same example here of adding n numbers in parallel. So my first question was name of the data across processors and communication values.

So name of the data that is this was my array A of i and this array was stored in some memory in a sequential system. When I chop the data across multiple processors, these two items where are they stored? Are they stored in registers? How are they copied from the main memory? Do they move to a local memory? Are they cached? Can they be only loaded in registers and so on. Okay. So there are many questions to be answered when we chop the data. Then communicating values. Okay. So for communicating values once this answer yellow answer is ready these two yellow circles, we need to communicate these two yellow circles to some processing element which can do the addition to generate the pink answer.

So this value has to be transmitted. I need to transmit this value from a particular processor. So suppose this was P5 and P6, these were two processors. So from P5 I need to send the value to P6 or vice versa. So whoever has taken the responsibility of adding we will have to transmit the local answer to that processor. So a communication is required to do this.

Then coordination and synchronization. Alright. So what does this mean? When the pink circle that is the pink answer will be generated only when the two yellows are ready. Okay. So I have yellow 1 and yellow 2 we add them to get pink 1. Okay. Suppose this is the structure. Now how do we know when to start the addition that is when is Y1 and Y2 ready? Are these values ready to be added? Who will tell this to us? Will processors P5 and P6 tell? or we have to find that out that is when is Y1 and Y2 ready? So all that that is called coordination and synchronization. We need to establish that before we can generate the pink answer. Okay.

Then you have to select the size of the processing element. In my particular example I had just need to add so I need an adder. Okay. And depending on the precision of the element I will need a 4 bit adder and all the way to 64 bit adder. Okay. So we will need an adder of one of these sizes depending on the data type. But if I want to do a multiplication I will need a multiplier.

If I want to do several complex arithmetic operations I might need a ALU. Even further for example you are doing some signal processing or something you will need a more complex code. Okay. So what type of processing node do you need? From a few bits ALU to or do you need a complete processor? And then selecting the number of nodes in the system. So which is the best way or how many cores do I need to solve this problem? 5, 10, 100? Okay so that decision has to be done. Alright. So we saw here that writing sequential programs is not sufficient.

You need to rethink and design parallel programs. But parallel programs do not come for free. You need to keep track of all these factors when you write parallel programs. And we will see in the future lectures how do we establish or give answers to these questions. Okay. Thank you.